REMARKS

Claims 1-26 are pending in the present application. Reconsideration of the claims is respectfully requested.

I. 35 U.S.C. § 102, Anticipation

The examiner has rejected claims 6-11, 17-22, and 25 under 35 U.S.C. § 102(e) as anticipated by Meltzer et al. (6,226,675). This rejection is respectfully traversed. In the current action, the examiner notes that

> *"In response to applicant's argument that the references fail to show certain features of applicant's invention, it is noted that the features upon which applicant relies (i.e., a dynamic translation program) is not recited in the rejected claim(s)."*[1]

It is respectfully submitted that the examiner has at least partially misunderstood applicant's arguments. Applicant has argued that the steps recited in the claims are necessary parts of dynamically translating an application. Meltzer does not perform the recited steps in the manner that they are claimed; <u>therefore</u> Meltzer is <u>neither</u> meeting the claimed limitations , <u>nor</u> is it performing a dynamic translation of an application. Instead, Meltzer is translating a document.

Revisiting the claims, exemplary Claim 6 recites

> *6. A method of generating a markup language file, the method comprising the computer-implemented steps of:*
> *executing an application program;*
> *parsing a document type definition file for a markup language;*
> *selecting an element defined in the document type definition file based on a routine called by the application program; and*
> *writing the selected element to a markup language file.*

This claim is directed to "generating a markup language file" and the writing step is clearly writing such a file. But this is not just any markup language file. The elements that have been selected to write to the markup language file are chosen from the DTD file, which is for a markup language according to the parsing step, but the elements are not chosen randomly. Rather, the elements are chosen based on a routine called by the application program. This can be seen to be a translation, not of the words of the

[1] Office Action of 12/18/02, page 2, item 3A

application file, but of the routines the application file is calling. The translation is dynamic because the source code for the application is not examined; rather, the application must execute so that the selecting step can see the routines it is calling. Only after the application program has called a routine can the selecting step perform its selection and the writing step write the selected element. Thus, although the words "dynamic translation" are never used in the claim, the steps perform a dynamic translation of the application program.

It should be noted that the contribution the application program brings to the generation of the markup language file is that while the application is being executed, the selecting step can choose elements based on routines that the application program calls. Generating the markup file based on what the application does is the essence of dynamically translating the application into markup language.[2] This relationship between the application and the chosen elements cannot be ignored in the claims, or the claim makes no sense.

The rejection of Claim 6 cites Meltzer as disclosing:

> *A method ... for generating a markup language file, comprising:*
> *executing an application program (Meltzer on col. 23, lines 17-60: teaches JAVA (application program));*
> *parsing a document type definition file for a markup language (Meltzer on col. 23, lines 38-60: teaches parsing XML DTD);*
> *selecting an element defined in the document type definition file based on a routine called by the application program (Meltzer on col. 23, lines 38-60): teaches element retrieved from XML DTD and col. 23, lines 17-60: teaches JAVA (application program); and*
> *writing the selected element to a markup language file (Meltzer on col. 23, lines 38-60: teaches producing an output by received XML element).*

Thus, it is asserted in this rejection that Meltzer discloses each and every limitation of this claim in column 23, lines 17-60, which are reproduced here.

[2] This is supported in the summary of the invention, which notes "*The present invention provides a method and apparatus for converting programs and source code files written in a programming language to equivalent markup language files. The conversion may be accomplished by a static process or by a dynamic process. ... In a dynamic process, the program is executed to generate the markup language file that corresponds to the source code file or presentation steps of the program. The application program is executed; a document type definition file for a markup language is provided as input; an element defined in the document type definition file is selected based on a routine called by the application program; and the selected element is written to a markup language file.*" - Application, page 4, entire page

"Furthermore, according to the present invention the applications that the listeners run need not be native XML functions, or native functions which match the format of the incoming document. Rather, these listeners may be JAVA functions, if the transaction process front end 304 is a JAVA interface, or may be functions which run according to a unique transaction processing architecture. In these cases, the objects would be transformed into the format required by the receiving application. When the application of the listener finishes, its output is then transformed back into the format of a document as specified by the business interface definition in the module 303. Thus, the translator 302 is coupled to the network interface 300 directly for supplying the composed documents as outputs.

The listeners coupled to the transaction processing front end may include listeners for input documents, listeners for specific elements of the input documents, and listeners for attributes stored in particular elements of the input document. This enables diverse and flexible implementations of transaction processes at the participant nodes for filtering and responding to incoming documents.

FIG. 4 illustrates a process of receiving and processing an incoming document for the system of FIG. 3. Thus, the process begins by receiving a document at the network interface (step 400). The parser identifies the document type (401) in response to the business interface definition. Using the business interface definition, which stores a DTD for the document in the XML format, the document is parsed (step 402). Next, the elements and attributes of the document are translated into the format of the host (step 403). In this example, the XML logic structures are translated into JAVA objects which carry the data of the XML element as well as methods associated with the data such as get and set functions. Next, the host objects are transferred to the host transaction processing front end (step 404). These objects are routed to processes in response to the events indicated by the parser and the translator. The processes which receive the elements of the document are executed and produce an output (step 405). The output is translated to the format of an output document as defined by the business interface definition (step 406). In this example, the translation proceeds from the form of a JAVA object to that of an XML document. Finally, the output document is transmitted to its destination through the network interface (step 407)."[3]

If we accept Java as the application program, as the rejection suggests, then Meltzer should disclose selecting an element in the DTD, <u>based</u> on a routine called by Java. But there is no discussion in Meltzer of routines called by Java, nor does Meltzer show the selection of an element in the DTD based on a routine called by Java.

[3] Meltzer, column 23, lines 7-60

Quite simply, the claim limitations are not met. Meltzer has some elements that superficially resemble the claimed limitations, but the resemblance falls apart on closer examination. While it is true that Meltzer shows Java, which can be considered an application program, Meltzer does not show the recited relationships between Java, routines called by Java, and elements in the DTD. Neither does Meltzer show the steps recited in the claim. To meet the claim limitations, Meltzer must show something that is monitoring the routines called by the application program (Java, according to the rejections) and selecting a markup language element based on the routines called. Meltzer does not show this.

If claims are to mean anything, they must be read carefully, with all their limitations taken into account. Not only must recited elements be present, but they must have the recited relationships to each other and they must act in the recited ways. Meltzer contains some of the elements of the claims, but they do not have the recited relationships to each other and they do not interact in the manner recited. Meltzer does not meet the requirements necessary to a 102 rejection, so this rejection is overcome.

II. 35 U.S.C. § 103, Obviousness

The examiner has rejected claims 1-5, 12-16, 23-24, and 26 under 35 U.S.C. §103(a) as being unpatentable over Meltzer et al. (6,226,675) in view of Day et al. (5,953,526). This rejection is also respectfully traversed.

Representative Claim 1 reads:

> *1. (Amended) A method of processing a source code statement written in a programming language, the method comprising the computer-implemented steps of:*
> *parsing a document type definition file for a markup language;*
> *parsing said source code statement from a source code file;*
> *selecting an element defined in the document type definition file based on an association between the element and an identifier of a routine in said source code statement; and*
> *writing the selected element to a markup language file.*

In a manner similar to Claim 6 discussed above, although this claim does not specifically recite "translating a source code document into a markup language", the steps it recites accomplish this end. In the two parsing steps, both the DTD for the markup language and the source code statement from the source code file are semantically broken down into their component parts. Using these component parts, elements from the DTD are found that have an association to (i.e., are translations of) source code statements. The associated elements are written to the markup language file to form the translation. However, as in the earlier rejection, the claim only makes sense if the relationship between the elements in the DTD and the routines in the source code statement is maintained. Without this association, the whole concept of the claimed method is nonfunctional and essentially random elements in a DTD can be written to the markup language file. Therefore, to maintain any sense to the claimed limitations, the combination of Meltzer and Day must also show an association between the DTD file and the source code statement being processed. But does it?

In the rejection, the examiner states:

> *"Regarding independent claims 1, 12, and 23, Meltzer discloses:*
> *A method ... for processing a source code statement written in a programming language (Meltzer on col. 23, lines 38-60; teaches JAVA object), comprising:*
> *parsing a document type definition file for a markup language (Meltzer on col. 23, lines 38-60; teaches parsing document type of XML format);*
> *selecting an element defined in the document type definition file (Meltzer on col. 3, lines 28-45; teaches data typing of elements within XML document type definition DTD) based on an association between the element and an identifier of a routine in said source code statement (Melter on col. 23 lines 38-60 and col. 79. lines 54-55: teaches selected JAVA objects to proceed with the translation into XML and association between parsed XML document with document type definition).*
> *Meltzer does not explicitly disclose "parsing a source code statement from a source code file". However, Day on col. 7, lines 24-50 and col. 8, lines 12-45: teaches parsing JAVA file to look for package statement."*[4]

It is submitted that the rejection is combining the two cited patents in a manner inconsistent with the method recited in the claims. The rejection would have us take the parsing of a source code statement from Day and insert that step into a portion of Meltzer, where there was previously no mention of parsing a source code statement. But, with the source code coming from the work of Day and the DTD being derived from the work of

[4] Office Action of 12/18/02, item 7, pages 6-7

Meltzer, there is no inherent connection between the source code and the DTD recited in the claims. As we previously mentioned, it is vital to the correct functioning of the recited method that the source code and DTD bear at least a partial correlation. Ignoring the necessity of this correlation between the source code and the DTD makes a mockery of the claims.

It is noted that in the response to applicant's previous amendment, the Examiner states that "*Meltzer on col. 79, lines 54-55: teaches parsing XML document (source code file) according to the document type definition which matches it (association).*" In response, it is submitted that the Examiner has relied on Day, not on Meltzer, for parsing a source code statement. Other portions of Meltzer may mention an association between a source code file and a DTD, but they are not the ones relied on in the rejection (nor do they appear to meet the necessary criteria to meet the claims). The Examiner's arguments are not congruent with the rejection.

It is therefore submitted that Meltzer and Day do not meet the claimed limitations and this rejection is overcome.

III. Conclusion

It is respectfully urged that the subject application is patentable over Meltzer *et al.* and Day and is now in condition for allowance.

The examiner is invited to call the undersigned at the below-listed telephone number if in the opinion of the examiner such a telephone conference would expedite or aid the prosecution and examination of this application.

DATE: January 16, 2003

Respectfully submitted,

Betty Formby

Betty Formby
Reg. No. 36,536
Carstens, Yee & Cahoon, LLP
P.O. Box 802334
Dallas, TX 75380
(972) 367-2001
Agent for Applicants